Creación e implementación de un protocolo de la Capa de Aplicación

Alma Rosa Páes Alcalá
Alejandro Valderrama Silva

Diciembre 9, 2019

Objetivo:

El objetivo de esta aplicación es llevar entretenimiento a gente que se quiere distraer de su cotidianidad a través de uno de los mejores temas de la historia: Pokemon. Pero no se queda ahí, nuestra aplicación también busca que los usuarios interesados puedan aprender un poco acerca de cómo diseñar y crear un protocolo TCP y maneras generales de ocupar bibliotecas de Python para ello. Esto es fácilmente lograble gracias a la documentación que proporcionamos y a nuestro código libre.

Una de las ventajas de esta aplicación es que es fácilmente adaptable a nuevas necesidades que surjan en el mercado ya que la estructura y la forma de comunicación no quedará obsoleta y puede aplicarse a distintas áreas sin problemas. Además esta estructura (base) puede ser utilizada en el desarrollo de nuevas aplicaciones que requieran el diseño de protocolos en TCP.

Aunado a esto, esta aplicación recopila la esencia de Pokemon; la cual consideramos que es capturar Pokemones sin importar sus características y así lograr conseguir el mejor Pokedex y convertirte en el mejor entrenador Pokemon. Que mejor que hacer esto con los Pokemones de la primera generación, los que inspiraron y generaron su éxito.

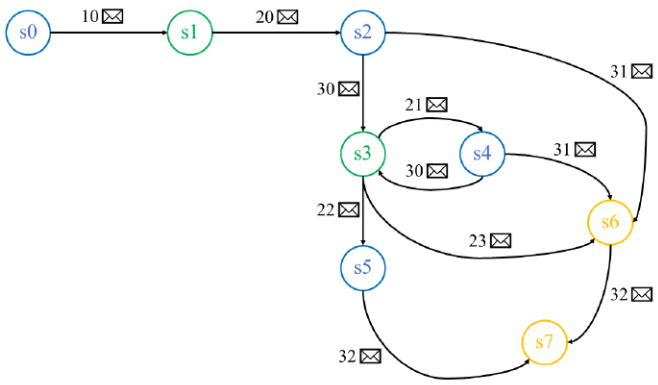
Repositorio:

Todo lo necesario para realizar lo mencionado en este documento, incluyendo instalar y jugar con la aplicación, se encuentran en la siguiente liga.

https://github.com/AlmaPaes/Proyecto2-Pokemon

Diseño del protocolo:

Como base del diseño del protocolo se tomo el autómata ya proporcionado, es decir:



Donde cada uno de los esta A los cuales los mensajes que le corresponden son los siguientes:

En común	
Código	Descripción
30	Sí.
31	No.
32	Terminando sesión.

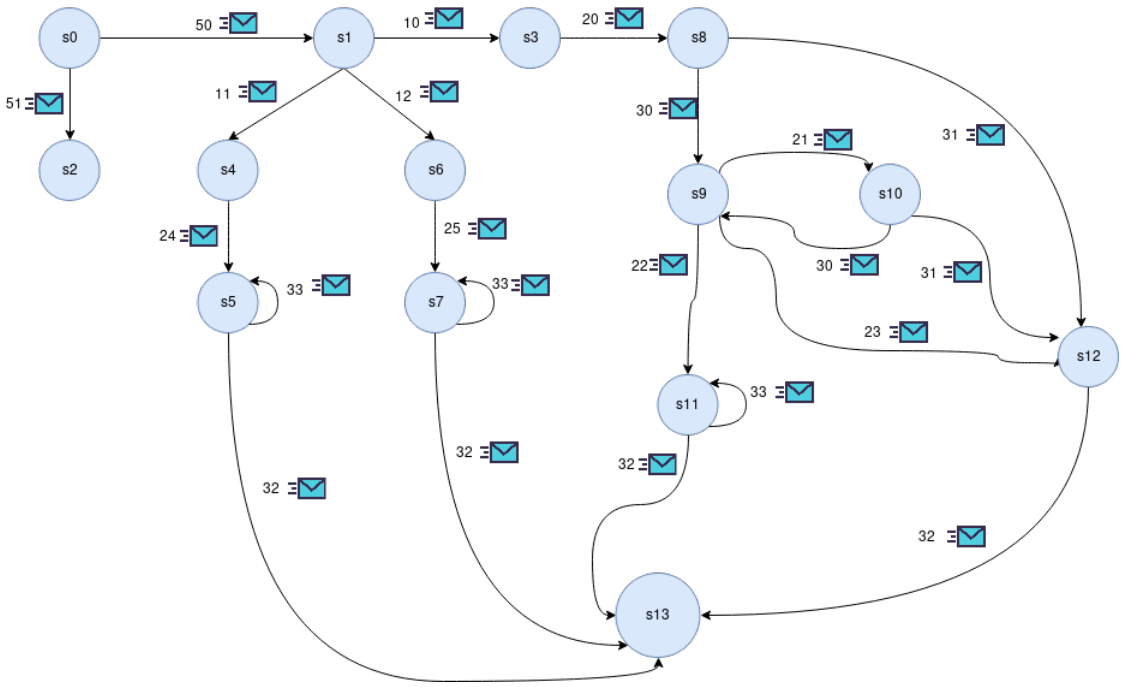
Servidor	
Código	Descripción
20	¿Capturar al Pokemon x?.
21	¿Intentar captura de nuevo? Quedan k intentos.
22	Envía Pokemon (imagen) capturado.
23	Número de intentos de captura agotados.

Cliente	
Código	Descripción
10	Solicitar al servidor por parte del cliente, un Pokemon para capturar.

Y los estados representaban lo siguiente:

Estados	
Código	Descripción
s0	Estado inicial, desde aquí comienza la conexión del protocolo de la capa de aplicación.
s1	Recibe solicitud del cliente, ofrece aleatoriamente un Pokemon para capturar.
s2	Indica si quiere capturar o no el Pokemon ofrecido.
s3	Inicia un contador con {n} como el máximo número de intentos. Aleatoriamente indica si se capturó al Pokemon o no.
s4	Da respuesta para reintentar captura de Pokemon.
s5	Se recibe Pokemon capturado (imagen).
s6	Terminando la sesión.
s7	Cierre de conexión.

Tomado en cuenta esta base, diseñamos el siguiente protocolo. Veamos primero el autómata de este nuevo protocolo:



Cuya tabla de transiciones es:

	10	11	12	20	21	22	23	24	25	30	31	32	33	40	41	50	51
s0																s1	s2
s1	s3	s4	s6														
s2																	
s3				s8													
s4								s5									
s5												s13	s5				
s6									s7								
s7												s13	s7				
s8										s9	s12						
s9					s10	s11	s12										
s10										s9	s12						
s11												s13	s11				
s12												s13					
s13																	

Para el cual agregamos algunos mensajes y mantuvimos los que ya existían, así quedando definidos de la siguiente manera:

En común	
Código	Descripción
30	Sí.
31	No.
32	Terminando sesión.
33	Recibido (Ack).

Servidor	
Código	Descripción
20	¿Capturar al Pokemon x?.
21	¿Intentar captura de nuevo? Quedan k intentos.
22	Envía Pokemon (imagen) capturado.
23	Número de intentos de captura agotados.
24	Envía Pokedex.
25	Envía catálogo.

Cliente	
Código	Descripción
10	Solicitar al servidor por parte del cliente, un Pokemon para capturar.
11	Solicitar al servidor por parte del cliente, ver su Pokedex.
12	Solicitar al servidor por parte del cliente, ver el catálogo.

Además agregamos dos nuevos grupos de códigos: 40 para errores y 50 para funcionalidades del login.

Errores	
Código	Descripción
50	Acceso permitido.
51	Acceso denegado.

Login	
Código	Descripción
40	Se rechaza la solicitud de conexión
41	Se cierra la conexión por timeout. Espera finalizada.

Como se modifico un poco el diseño de los mensajes y el flujo de comunicación entre el servidor y el cliente, los estados quedaron definidos de la siguiente manera:

Estados	
Código	Descripción
s0	Estado inicial, desde aquí comienza la conexión del protocolo de la capa de aplicación. Se solicitan credenciales para ingresar.
s1	Acceso concedido. Entrada a la apliacación. Se indica si se desea atrapar un Pokemon o revisar el catálogo y el Pokedex
s2	Acceso denegado.
s3	Se recibe una solicitud para capturar un Pokemon, y manda el id del Pokemon a capturar.
s4	Se recibe una solicitud para ver el Pokedex.
s5	El servidor comienza a transferir el Pokedex al cliente, muestra el Pokedex al usuario
s6	Se recibe una solicitud para ver el catálogo de Pokemones.
s7	El servidor comienza a transferir el catalogo al cliente, muestra el catálogo de Pokemones al usuario.
s8	Indica si quiere capturar o no el Pokemon ofrecido
s9	Se inicia un contador con el máximo número de intentos, e indica aleatoriamente si se capturó el Pokemon o no
s10	Da respuesta para reintentar captura de Pokemon
s11	El cliente recibe la imagen del servidor, se muestra imagen del Pokemon capturado
s12	Terminando la sesión.
s13	Cierre de conexión.

Finalmente, los mensajes se conforman de la siguiente manera.
Todos los mensajes para los códigos 10,11,12,23,30,31,32,33,40,41,50,51 estarán conformados por un byte, es decir:

code
1 byte

El mensaje 20, de la siguiente manera:

code	idPokemon
1 byte	1 byte

El mensaje 21, de la siguiente manera:

code	idPokemon	numAttemps
1 byte	1 byte	1 bytes

El mensaje 22, de la siguiente manera:

code	idPokemon	imageSize	image
1 byte	1 byte	4 bytes	k bytes

El mensaje 24, de la siguiente manera:

code	pokedexSize	Pokedex
1 byte	4 bytes	k bytes

El mensaje 25, de la siguiente manera:

code	catálogoSize	Catálogo
1 byte	4 bytes	k bytes

¿Cómo usar?

Es importante señalar que primero inicializamos el servidor, y después los clientes pueden iniciar una conexión. Entonces, para cada uno de éstos se debe hacer lo siguiente:

- **Para el servidor**
  - Si es la primera vez usando el servidor del juego, por favor dirigirse a la carpeta `./Instalaciones` y ejecutar `./make_servidor`

- En caso contrario: en una terminal, nos situamos en la ubicación del archivo *pokemonServer.py*
- No necesitamos parámetros extra para ejecutar el servidor. `./pokemonServer.py`

- **Para el cliente**
  - Si es la primera vez usando un cliente del juego, por favor dirigirse a la carpeta `./Instalaciones` y ejecutar `./make_cliente`
  - En una terminal, nos situamos en la ubicación del archivo *pokemonClient.py*
  - Este programa recibe como parámetros iniciales la dirección IP a través de la cual se quiere conectar, y el puerto. Por lo tanto, ejecutamos de la siguiente manera: `./pokemonClient.py <IP> <port>`

A continuación mostraremos, a forma de tutorial, como ejecutar tanto el servidor como el cliente. Además mostraremos como, por parte del cliente, interactucar con la aplicación.

Paso a Paso:

Como ya vimos, primero debemos de inicializar el servidor; para lo cual hacemos lo siguiente:

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonServer.py
Socket created
Socket now listening
```

A continuación podemos ejecutar el cliente[1] e ingresar nuestras credenciales para comenzar el juego.

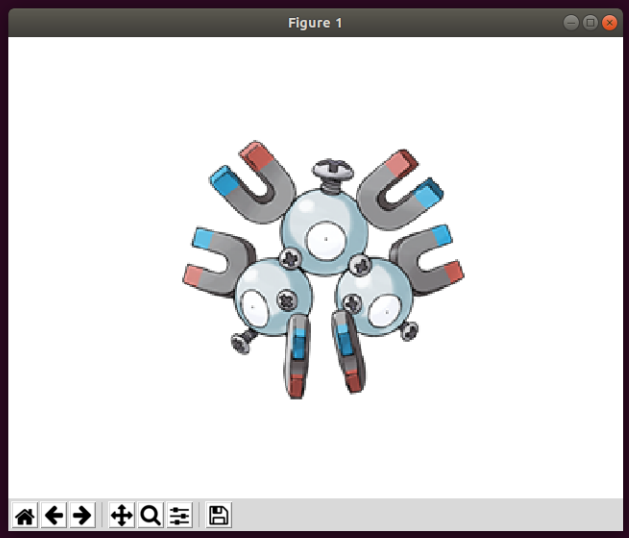
```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Paulo
Ingrese la contraseña
 >>
                _
  _ __   ___ | | _____ _ __ ___   ___  _ __
 | '_ \ / _ \| |/ / _ \ '_ ` _ \ / _ \| '_ \
 | |_) | (_) |   <  __/ | | | | | (_) | | | |
 | .__/ \___/|_|\_\___|_| |_| |_|\___/|_| |_|
 |_|
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
 >>
```

Posteriormente podemos escoger cualquiera de las tres opciones para realizar alguna acción respecto al juego, o podemos elegir salir de la aplicación. Mostraremos que hacer en cada una de las opciones.

Veamos, primero, que hacer en caso de que deseemos atrapar un Pokemon. Para esto debemos, necesariamente, ingresar la opción *P*. Al hacer esto la aplicación nos ofrecerá un Pokemon para capturar, nosotros podemos escoger si capturarlo o no. En caso de que lo capturemos nos mandará la imagen del Pokemon capturado; es importante aclarar que antes de seguir con el flujo del programa debemos cerrar el visor de la imagen, podemos guardarla si así lo deseamos. En otro caso podemos volver a intentar capturarlo (siempre y cuando tengamos intentos disponibles).

[1] En realidad se pueden ejecutar varios clientes a la vez. El servidor puede atender hasta diez simultáneamente

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Paulo
Ingrese la contraseña
 >>
              _
  _ __   ___ | | _____ _ __ ___   ___  _ __
 | '_ \ / _ \| |/ / _ \ '_ ` _ \ / _ \| '_ \
 | |_) | (_) |   <  __/ | | | | | (_) | | | |
 | .__/ \___/|_|\_\___|_| |_| |_|\___/|_| |_|
 |_|
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
 >> P
¿Capturar al Pokemon Magneton?
Sí [S] o No [N]
 >> S
¿Intentar captura de nuevo? Quedan 4 intentos
Sí [S] o No [N]
 >> S
¿Intentar captura de nuevo? Quedan 3 intentos
Sí [S] o No [N]
 >> S
¿Intentar captura de nuevo? Quedan 2 intentos
Sí [S] o No [N]
 >> S
Capturaste al pokemon...
```



Si no lo capturamos, resultaría:

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Alma
Ingrese la contraseña
 >>
              _
  _ __   ___ | | _____ _ __ ___   ___  _ __
 | '_ \ / _ \| |/ / _ \ '_ ` _ \ / _ \| '_ \
 | |_) | (_) |   <  __/ | | | | | (_) | | | |
 | .__/ \___/|_|\_\___|_| |_| |_|\___/|_| |_|
 |_|
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
 >> P
¿Capturar al Pokemon Poliwag?
Sí [S] o No [N]
 >> S
¿Intentar captura de nuevo? Quedan 1 intentos
Sí [S] o No [N]
 >> S
Te quedaste sin intentos :(
Terminando conexión...
```

Luego, podemos revisar el Pokedex[2] de la siguiente manera:

[2]Es decir, podemos consultar los Pokemones que hemos capturado

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Paulo
Ingrese la contraseña
 >>
              _
  _ __   ___ | | _____ _ __ ___    ___  _ __
 | '_ \ / _ \| |/ / _ \ '_ ` _ \ / _ \| '_ \
 | |_) | (_) |   <  __/ | | | | | (_) | | | |
 | .__/ \___/|_|\_\___|_| |_| |_|\___/|_| |_|
 |_|
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
 >> X
Mostrando Pokedex...
Bellsprout, Koffing,
Farfetchd, Drowzee,
```

Finalmente, podemos revisar el catálogo[3] de todos los Pokemones disponibles en la aplicación.

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Paulo
Ingrese la contraseña
 >>
              _
  _ __   ___ | | _____ _ __ ___    ___  _ __
 | '_ \ / _ \| |/ / _ \ '_ ` _ \ / _ \| '_ \
 | |_) | (_) |   <  __/ | | | | | (_) | | | |
 | .__/ \___/|_|\_\___|_| |_| |_|\___/|_| |_|
 |_|
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
 >> C
Mostrando catálogo...
Bulbasaur, Ivysaur, Venusaur, Charmander, Charmeleon, Charizard,
Squirtle, Wartortle, Blastoise, Caterpie, Metapod, Butterfree,
Weedle, Kakuna, Beedrill, Pidgey, Pidgeotto, Pidgeot,
Rattata, Raticate, Spearow, Fearow, Ekans, Arbok,
Pikachu, Raichu, Sandshrew, Sandslash, Nidoran (female), Nidorina,
Nidoqueen, Nidoran (male), Nidorino, Nidoking, Clefairy, Clefable,
Vulpix, Ninetales, Jigglypuff, Wigglytuff, Zubat, Golbat,
Oddish, Gloom, Vileplume, Paras, Parasect, Venonat,
Venomoth, Diglett, Dugtrio, Meowth, Persian, Psyduck,
Golduck, Mankey, Primeape, Growlithe, Arcanine, Poliwag,
Poliwhirl, Poliwhrath, Abra, Kadabra, Alakazam, Machop,
Machoke, Machamp, Bellsprout, Weepinbell, Victreebel, Tentacool,
Tentacruel, Geodude, Graveler, Golem, Ponyta, Rapidash,
Slowpoke, Slowbro, Magnemite, Magneton, Farfetchd, Doduo,
Dodrio, Seel, Dewgong, Grimer, Mulk, Shellder,
Cloyster, Gastly, Haunter, Gengar, Onix, Drowzee,
Hypno, Krabby, Kingler, Voltorb, Electrode, Exeggcute,
Exeggutor, Cubone, Marowak, Hitmonlee, Hitmonchan, Lickitung,
Koffing, Weezing, Rhyhorn, Rhydon, Chansey, Tangela,
Kangaskhan, Horsea, Seadra, Goldeen, Seaking, Staryu,
Starmie, Mr. Mime, Scyther, Jynx, Electabuzz, Magmar,
Pinsir, Tauros, Magikarp, Gyarados, Lapras, Ditto,
Eevee, Vaporeon, Jolteon, Flareon, Polygon, Omanyte,
Omastar, Kabuto, Kabutops, Aerodactyl, Snorlax, Articuno,
Zapdos, Moltres, Dratini, Dragonair, Dragonite, Mewtwo,
```

Errores comúnes: En caso de haber inactividad en la aplicación, esta generará errores porque se ha excedido el tiempo de espera de la comunicación (timeouts). Lo cual resultará en que la conexión entre el cliente y el servidor se termine.
A continuación mostramos algunos casos en los que puede resultar este error.

[3]Esta funcionalidad puede tardar unos momentos

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Paulo
Ingrese la contraseña
 >>
Tiempo de respuesta excedido: 10 segundos
Terminando conexión...
```

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Paulo
Ingrese la contraseña
 >>
              _
  _ __   ___ | | _____ _ __ ___   ___  _ __
 | '_ \ / _ \| |/ / _ \ '_ ` _ \ / _ \| '_ \
 | |_) | (_) |   <  __/ | | | | | (_) | | | |
 | .__/ \___/|_|\_\___|_| |_| |_|\___/|_| |_|
 |_|
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
 >> P
Tiempo de respuesta excedido: 10 segundos
Terminando conexión...
```

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
 >> Paulo
Ingrese la contraseña
 >>
              _
  _ __   ___ | | _____ _ __ ___   ___  _ __
 | '_ \ / _ \| |/ / _ \ '_ ` _ \ / _ \| '_ \
 | |_) | (_) |   <  __/ | | | | | (_) | | | |
 | .__/ \___/|_|\_\___|_| |_| |_|\___/|_| |_|
 |_|
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
 >> P
¿Capturar al Pokemon Kakuna?
Sí [S] o No [N]
 >> S
¿Intentar captura de nuevo? Quedan 3 intentos
Sí [S] o No [N]
 >> S
Tiempo de respuesta excedido: 10 segundos
Terminando conexión...
```

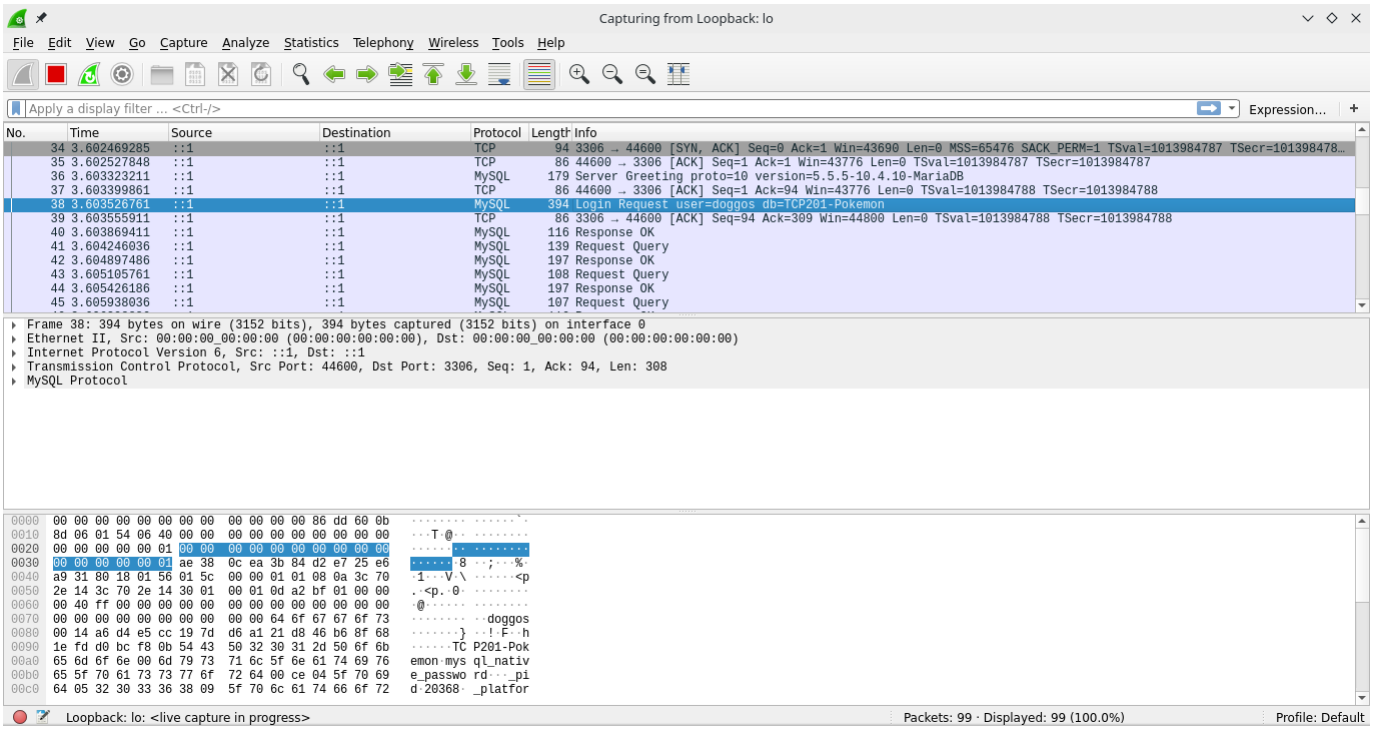
Por el lado del servidor veremos que aunque se muera la conexión éste seguirá escuchando y atendiendo peticiones. Por lo cual sólo veríamos algo como:

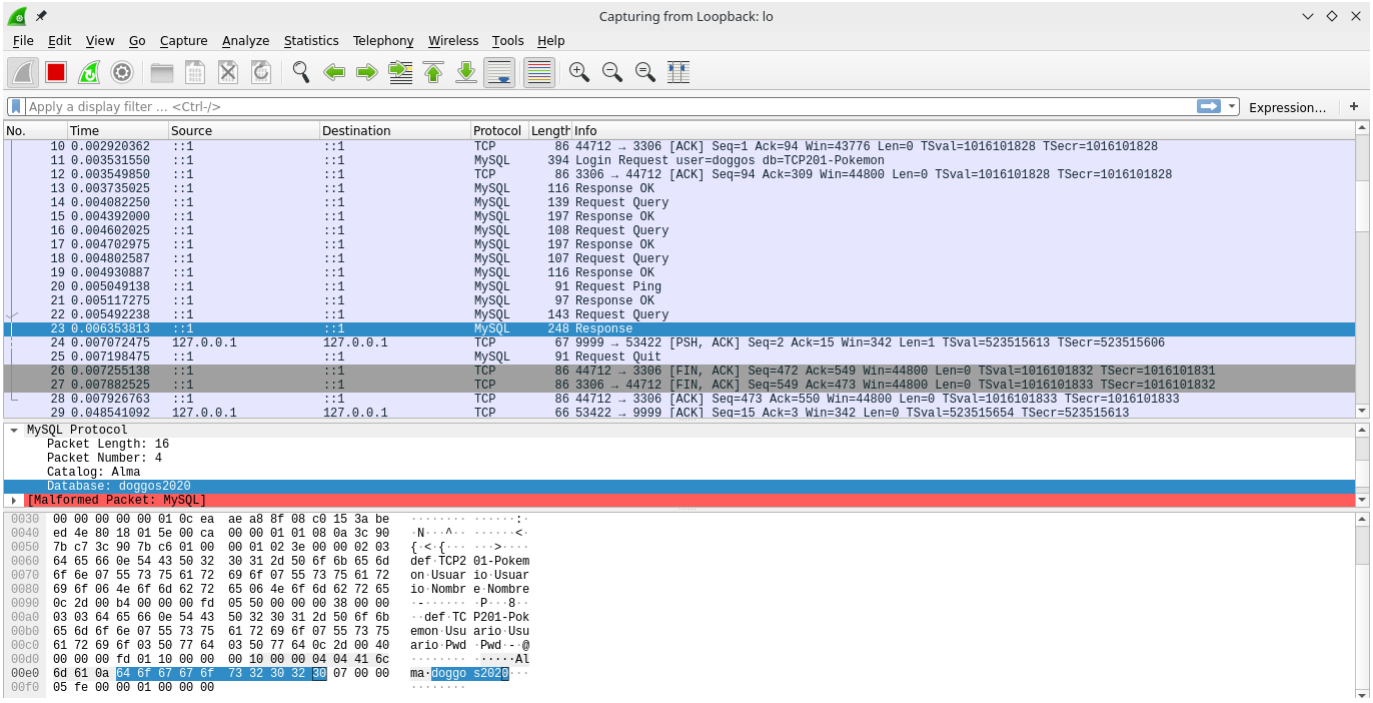
```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonServer.py 127.0.0.1
Socket created
Socket now listening
Connected with 127.0.0.1:56332
Tiempo de respuesta excedido: 10 segundos
```

Observación de su funcionamiento mediante capturas de Wireshark

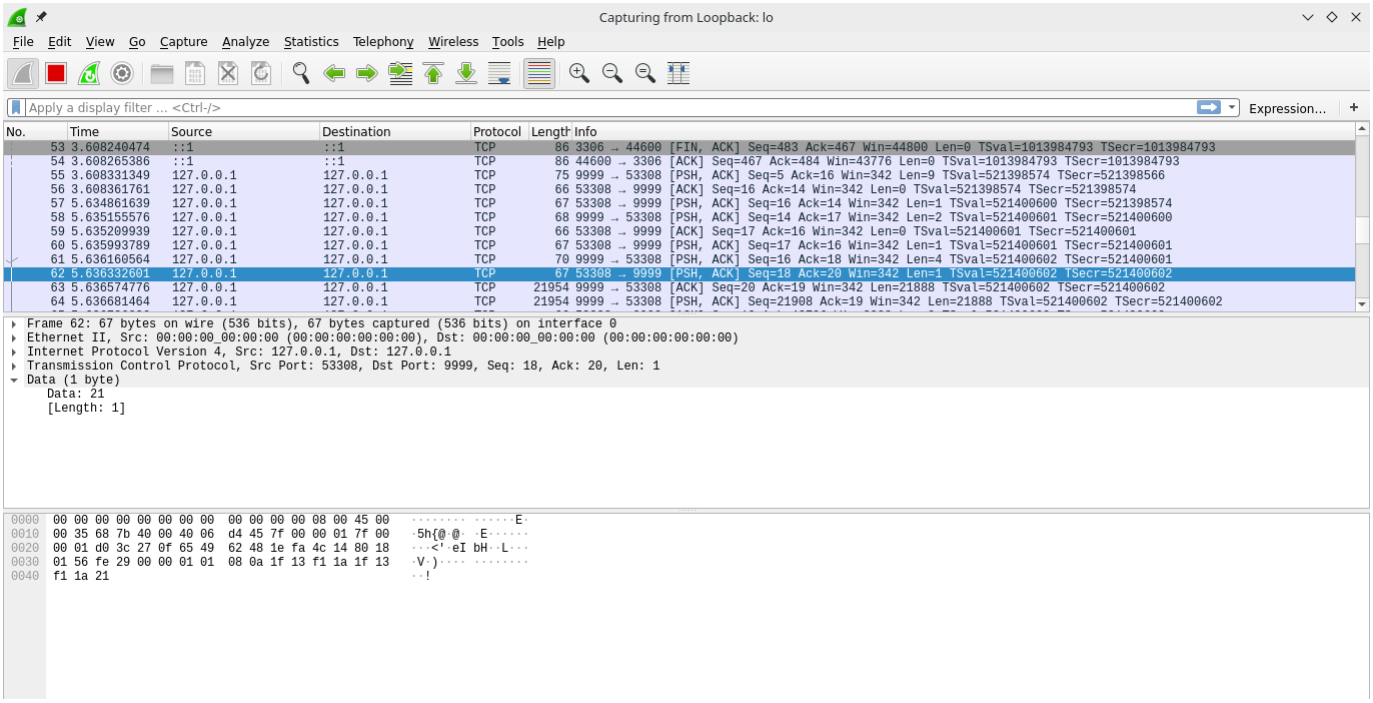
Profundizando en el funcionamiento interno del protocolo, se mostrarán lecturas capturadas con el sniffer *Wireshark* en el proceso de las distintas funciones de la aplicación.

Veamos el envío de mensajes al momento de ingresar a la aplicación, donde vemos interacción con la base de datos en MySQL. Notemos que en el área de datos se visualizan datos de la conexión y el query mandado:

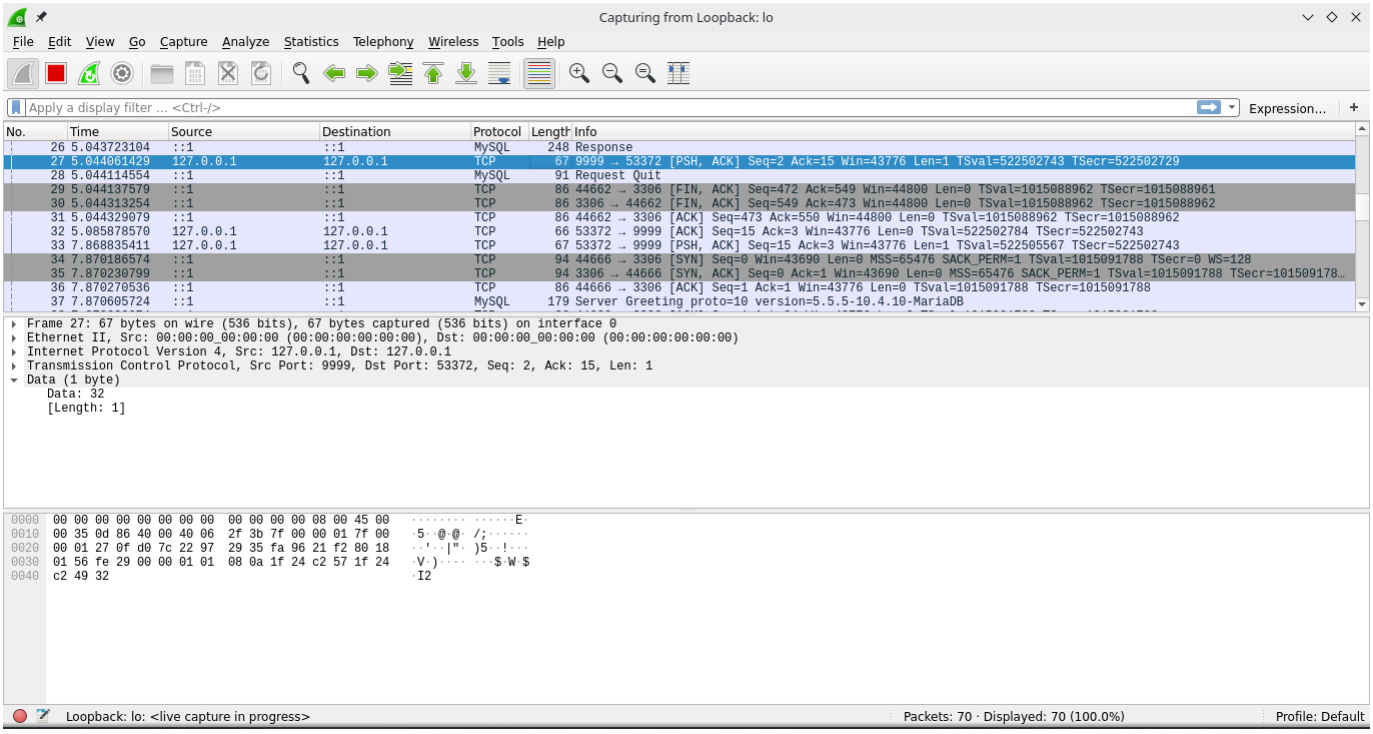

Capturing from Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
34 3.602469285 ::1 ::1 TCP 94 3306 → 44600 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=1013984787 TSecr=101398478…
35 3.602527848 ::1 ::1 TCP 86 44600 → 3306 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=1013984787 TSecr=1013984787
36 3.603323211 ::1 ::1 MySQL 179 Server Greeting proto=10 version=5.5.5-10.4.10-MariaDB
37 3.603399861 ::1 ::1 TCP 86 44600 → 3306 [ACK] Seq=1 Ack=94 Win=43776 Len=0 TSval=1013984788 TSecr=1013984788
38 3.603526761 ::1 ::1 MySQL 394 Login Request user=doggos db=TCP201-Pokemon
39 3.603555911 ::1 ::1 TCP 86 3306 → 44600 [ACK] Seq=94 Ack=309 Win=44800 Len=0 TSval=1013984788 TSecr=1013984788
40 3.603869411 ::1 ::1 MySQL 116 Response OK
41 3.604246036 ::1 ::1 MySQL 139 Request Query
42 3.604897486 ::1 ::1 MySQL 197 Response OK
43 3.605105761 ::1 ::1 MySQL 108 Request Query
44 3.605426186 ::1 ::1 MySQL 197 Response OK
45 3.605938036 ::1 ::1 MySQL 107 Request Query
Frame 38: 394 bytes on wire (3152 bits), 394 bytes captured (3152 bits) on interface 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 6, Src: ::1, Dst: ::1
Transmission Control Protocol, Src Port: 44600, Dst Port: 3306, Seq: 1, Ack: 94, Len: 308
MySQL Protocol
0000 00 00 00 00 00 00 00 00 00 00 00 00 86 dd 60 0b
0010 8d 06 01 54 06 40 00 00 00 00 00 00 00 00 00 00
0020 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 01 ae 38 0c ea 3b 84 d2 e7 25 e6
0040 a9 31 80 18 01 56 01 5c 00 00 01 01 08 0a 3c 70
0050 2e 14 3c 70 2e 14 30 01 00 01 0d a2 bf 01 00 00
0060 00 40 ff 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 64 6f 67 67 6f 73
0080 00 14 a6 d4 e5 cc 19 7d d6 a1 21 d8 46 b6 8f 68
0090 1e fd d0 bc f8 0b 54 43 50 32 30 31 2d 50 6f 6b
00a0 65 6d 6f 6e 00 6d 79 73 71 6c 5f 6e 61 74 69 76
00b0 65 5f 70 61 73 73 77 6f 72 64 00 ce 04 5f 70 69
00c0 64 05 32 30 33 36 38 09 5f 70 6c 61 74 66 6f 72
Loopback: lo: <live capture in progress>
Packets: 99 · Displayed: 99 (100.0%)
Profile: Default


Capturing from Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
10 0.002920362 ::1 ::1 TCP 86 44712 → 3306 [ACK] Seq=1 Ack=94 Win=43776 Len=0 TSval=1016101828 TSecr=1016101828
11 0.003531550 ::1 ::1 MySQL 394 Login Request user=doggos db=TCP201-Pokemon
12 0.003549850 ::1 ::1 TCP 86 3306 → 44712 [ACK] Seq=94 Ack=309 Win=44800 Len=0 TSval=1016101828 TSecr=1016101828
13 0.003735025 ::1 ::1 MySQL 116 Response OK
14 0.004082250 ::1 ::1 MySQL 139 Request Query
15 0.004392000 ::1 ::1 MySQL 197 Response OK
16 0.004602025 ::1 ::1 MySQL 108 Request Query
17 0.004702975 ::1 ::1 MySQL 197 Response OK
18 0.004802587 ::1 ::1 MySQL 107 Request Query
19 0.004930887 ::1 ::1 MySQL 116 Response OK
20 0.005049138 ::1 ::1 MySQL 91 Request Ping
21 0.005117275 ::1 ::1 MySQL 97 Response OK
22 0.005492238 ::1 ::1 MySQL 143 Request Query
23 0.006353813 ::1 ::1 MySQL 248 Response
24 0.007072475 127.0.0.1 127.0.0.1 TCP 67 9999 → 53422 [PSH, ACK] Seq=2 Ack=15 Win=342 Len=1 TSval=523515613 TSecr=523515606
25 0.007198475 ::1 ::1 MySQL 91 Request Quit
26 0.007255138 ::1 ::1 TCP 86 44712 → 3306 [FIN, ACK] Seq=472 Ack=549 Win=44800 Len=0 TSval=1016101832 TSecr=1016101831
27 0.007882525 ::1 ::1 TCP 86 3306 → 44712 [FIN, ACK] Seq=549 Ack=473 Win=44800 Len=0 TSval=1016101833 TSecr=1016101832
28 0.007926763 ::1 ::1 TCP 86 44712 → 3306 [ACK] Seq=473 Ack=550 Win=44800 Len=0 TSval=1016101833 TSecr=1016101833
29 0.048541092 127.0.0.1 127.0.0.1 TCP 66 53422 → 9999 [ACK] Seq=15 Ack=3 Win=342 Len=0 TSval=523515654 TSecr=523515613
MySQL Protocol
Packet Length: 16
Packet Number: 4
Catalog: Alma
Database: doggos2020
[Malformed Packet: MySQL]
0030 00 00 00 00 00 01 0c ea ae a8 8f 08 c0 15 3a be
0040 ed 4e 80 18 01 5e 00 ca 00 00 01 01 08 0a 3c 90
0050 7b c7 3c 90 7b c6 01 00 00 01 02 3e 00 00 02 03
0060 64 65 66 0e 54 43 50 32 30 31 2d 50 6f 6b 65 6d
0070 6f 6e 07 55 73 75 61 72 69 6f 07 55 73 75 61 72
0080 69 6f 06 4e 6f 6d 62 72 65 06 4e 6f 6d 62 72 65
0090 0c 2d 00 b4 00 00 00 fd 05 50 00 00 00 38 00 00
00a0 03 03 64 65 66 0e 54 43 50 32 30 31 2d 50 6f 6b
00b0 65 6d 6f 6e 07 55 73 75 61 72 69 6f 07 55 73 75
00c0 61 72 69 6f 03 50 77 64 03 50 77 64 0c 2d 00 40
00d0 00 00 00 fd 01 10 00 00 00 10 00 00 04 04 41 6c
00e0 6d 61 0a 64 6f 67 67 6f 73 32 30 32 30 07 00 00
00f0 05 fe 00 00 01 00 00 00

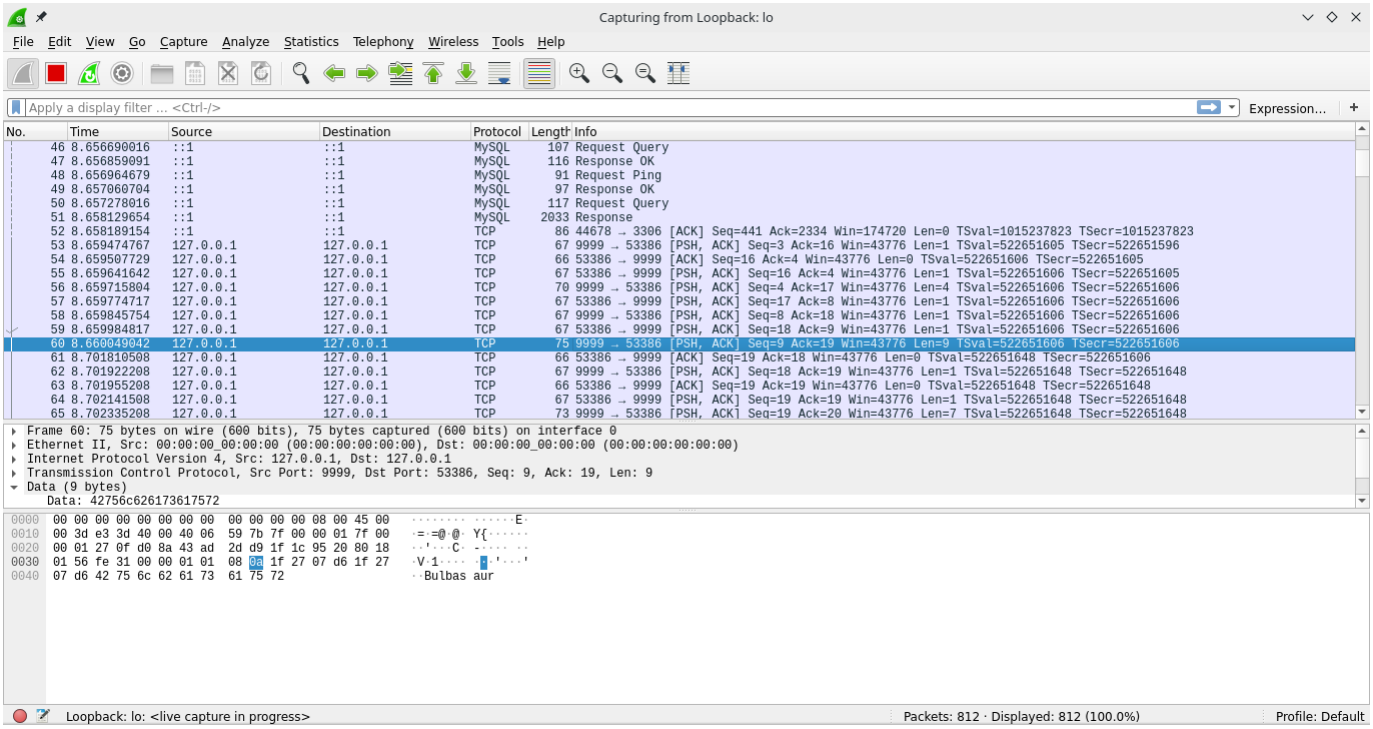
Mensajes enviados durante la captura de un Pokemon



Mensajes enviados durante la consulta del Pokedex



Mensajes enviados durante la consulta del catálogo de Pokemones



Funciones usadas en la programación:

Para revisar la lista de la funciones, ver el apartado de la documentación de las mismas al final del documento.

CAPÍTULO 2

Programas involucrados

2.1 Cliente Pokemon Go!

Implementación de un cliente para el juego Pokemon Go! e interactúa directamente con el usuario

pokemonClient.**cerrarSesion**(*soc*)
Cierre normal de sesión del usuario.

Parámetros **soc** (*Socket*) – Socket de la conexión

Devuelve Nada

pokemonClient.**displayCatalogo**(*catalogo*)

Imprime en pantalla el catálogo de Pokemones disponibles de manera «amigable».

Parámetros **catalogo** (*List of String*) – Catalogo de Pokemones

Devuelve Nada

pokemonClient.**displayPokedex**(*pokedex*)

Imprime en pantalla el Pokedex de manera «amigable».

Parámetros **pokedex** (*List of String*) – Pokedex de Pokemones

Devuelve Nada

pokemonClient.**login**(*soc*)
Transfiere los datos al servidor para validar el acceso, y cierra el programa si los datos no son válidos.

Parámetros **soc** (*Socket*) – Socket de la conexión

Devuelve Nada

pokemonClient.**main**()
Función principal

pokemonClient.**muestraCatalogo**(*soc*)

Le muestra el catálogo disponible de Pokemones al usuario.

Parámetros **soc** (*Socket*) – Socket de la conexión

Devuelve Nada

pokemonClient.**muestraPokedex**(*soc*)

Muestra el Pokedex del usuario que solicita esta acción al usuario.

Parámetros **soc** (*Socket*) – Socket de la conexión

Devuelve Nada

pokemonClient.**muestraPokemon**(*bytes*)
Despliega el pokemon asignado.

Parámetros **bytes** (*bytearray*) – bytes de la imagen del pokemon a desplegar

Devuelve Nada

pokemonClient.**playPokemon**(*soc*)
Permite que el usuario juegue Pokemon Go.

Parámetros **soc** (*Socket*) – Socket de la conexión

Devuelve Nada

pokemonClient.**printPokemon**()
Imprime en pantalla el logo Pokemon :returns: Nada

pokemonClient.**terminarConTimeout**(*soc*)

Termina la conexión pues el tiempo de espera de la respuesta del Servidor ha excedido.

Parámetros **soc** (*Socket*) – Socket de la conexión

Devuelve Nada

pokemonClient.**terminarConexion**()

Termina la conexion pues el Servidor notifica que el tiempo de espera ha excedido.

Param Nada

Devuelve Nada

2.2 Servidor Pokemon Go!

Implementación de un servidor para el juego Pokemon Go!

pokemonServer.**avisoTimeout**(*connection*)
Manda el mensaje de cierre de sesión al cliente por tiempo de espera excedido.

Parámetros **connection** (*Conexión*) – Conexión entre el cliente y el servidor

Devuelve Nada

pokemonServer.**cerrarSesion**(*connection*)
Cierre de sesión entre el servidor y el cliente al cual le pertenece la conexión.

Parámetros **connection** (*Conexión*) – Conexión entre el cliente y el servidor

Devuelve Nada

pokemonServer.**clientThread**(*connection*, *ip*, *port*, *max_buffer_size=5120*)
Manejador del hilo que sostiene la conexión entre el servidor y un cliente

Parámetros

- **connection** (*Conexión*) – Conexión entre el servidor y el cliente que abrió el hilo
- **ip** (*String*) – Dirección IP de la conexión
- **port** (*Integer*) – Puerto a través del cual el servidor mantiene la conexión con el cliente
- **max_buffer_size** (*Integer*) – Número máximo de bytes que puede recibir en un paquete del cliente

Devuelve Nada

pokemonServer.**getNombrePokemon**(*idPokemon*)
Regresa el nombre del pokemon dado su id.

Parámetros **idPokemon** (*Integer*) – Id del Pokemon a capturado

Devuelve String

pokemonServer.**giveAccess**(*connection*, *max_buffer_size=5120*)
Autentifica a usuarios registrados y proporciona acceso a la ejecución de la aplicación

Parámetros

- **connection** (*Conexión*) – Conexión entre el servidor y el cliente que abrió el hilo
- **max_buffer_size** (*Integer*) – Número máximo de bytes que puede recibir en un paquete del cliente

Devuelve Integer, String -> Valor que representa la correctud del accesso; Cadena que reprensenta al usuario activo.

pokemonServer.**guardaEnPokedex**(*idPokemon*, *user*)

Guarda el pokemon capturado en el pokedex del usuario.

Parámetros

- **idPokemon** (*Integer*) – Id del Pokemon a capturado
- **user** (*String*) – Usuario que capturo

Devuelve Nada

pokemonServer.**main**()
Función principal.

pokemonServer.**muestraCatalogo**(*connection*)
Muestra el catalogo.

Parámetros **connection** (*Conexión*) – Conexión entre el servidor y el cliente

Devuelve Nada

pokemonServer.**muestraPokedex**(*connection*, *user*)
Muestra el Pokedex del usuario.

Parámetros

- **connection** (*Conexión*) – Conexión entre el servidor y el cliente
- **user** (*String*) – Usuario que capturo

Devuelve Nada

pokemonServer.**playPokemonGo**(*connection*, *user*)
Método que simula el comportamiento del juego Pokemon Go.

Parámetros **connection** (*Conexión*) – Conexión entre el servidor y el cliente

Devuelve Nada

pokemonServer.**start_server**()
Inicialización del servidor

Parámetros **ip_dir** (*String*) – Dirección IP del socket al cual se va conectar el servidor

Devuelve Nada

pokemonServer.**terminarConexion**()

Termina la conexion pues el Cliente notifica que el tiempo de espera ha excedido.

Param Nada

Devuelve Nada